INSURANCE SUBSIDY METHODOLOGY

ver. 1.02

PODPŮRNÝ A GARANČNÍ ROLNICKÝ A LESNICKÝ FOND, a.s.
Praha 8 - Karlín, Sokolovská 394/17, PSČ 186 00, IČ 49241494
společnost zapsaná v obchodním rejstříku vedeném Městským soudem v Praze, v oddílu B, vložce č. 2130

Historie dokumentu

Verze	Datum	Autor	Popis
0.1	28.02.2023	Michal Valach	Initial version
1.0	03.03.2023	Michal Valach	Final version
1.0.1	30.03.2023	Michal Valach	Minor description changes
1.0.2	16.08.2023	Michal Valach	Final release version

Obsah

1 Introduction

Purpose of this document is to specify how and when to use service specified in document insurance_subsidy_sesp_v1.0.1.docx.

1.1 Related documents

The documents referenced in the table below are integrated parts of this document.

Table 1: Related documents

ID	DOCUMENT	COMMENT
1	insurance_subsidy_sesp_v1.0.1.docx	Service specification

1.2 Referenced documents

The documents referenced in the table below are relevant inputs to this document.

Table 2: Referenced documents

ID	DOCUMENT	COMMENT
1	insurance_subsidy_v1.0.wsdl	Service specification
2	insurance_subsidy_v1.0.xsd	XML Schema
3	common_v1.0.xsd	Common XML schema

2 Service introduction

Table 3: Service introduction

SERVICE NAME	**InsuranceSubsidy**
VERSION	1.0
DESCRIPTION	Service receives insurance payment status for subsidies
DOCUMENT MASTER	PGRLF

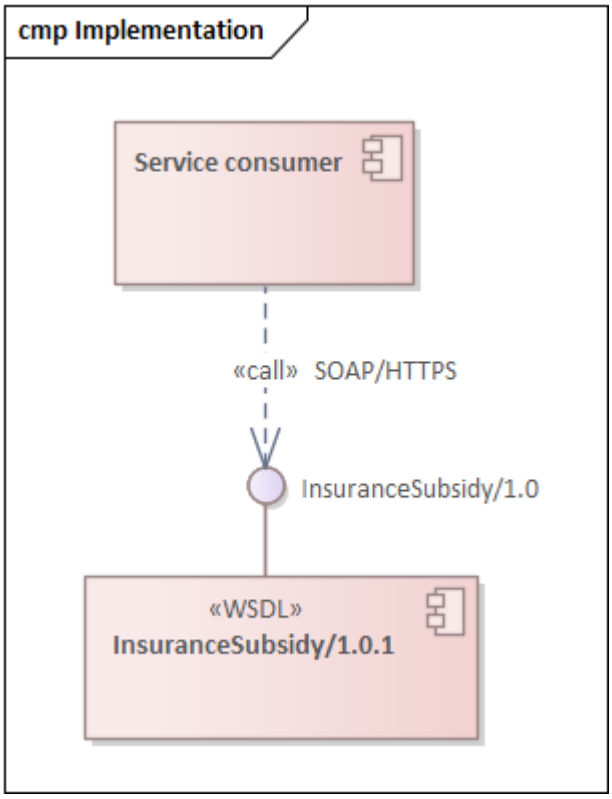
3 Interface description

3.1 Overview of interface

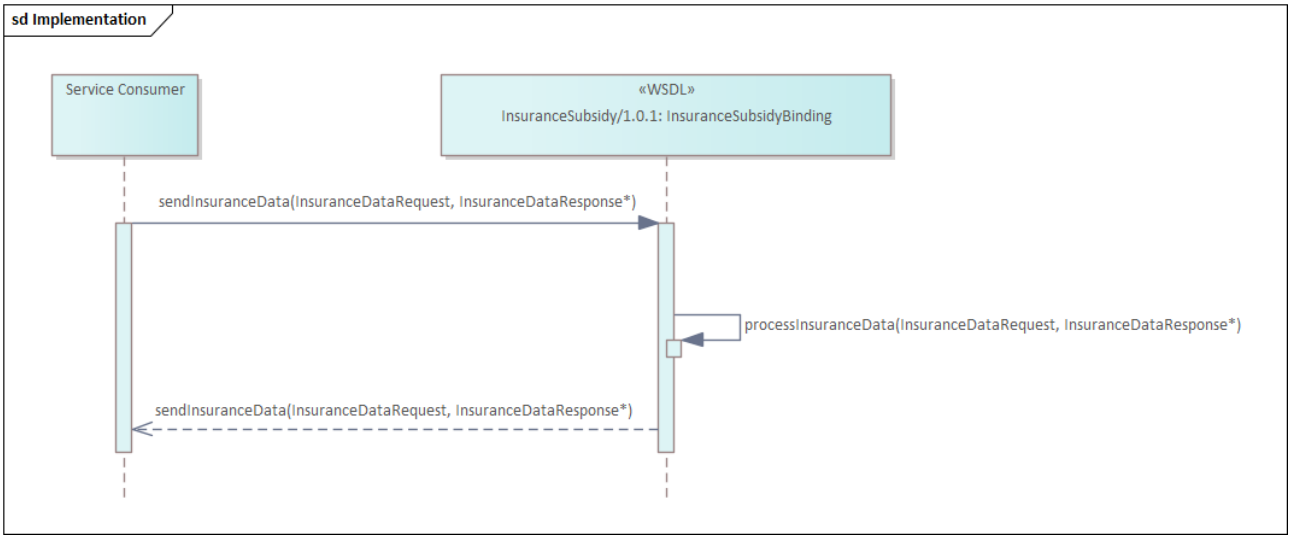
Interface enables insurance companies to send information about insurance and its payments, eventually insurance updates or cancellation.

3.2 Communication overview

3.2.1 Component diagram



3.2.2 Sequence diagram



3.3 Connected systems

Table 4: Connected systems

<table>
<tr><th>SYSTEM NAME</th><th>USAGE DETAILS</th></tr>
<tr><td>Pojišťovna</td><td>
<table>
<tr><th>OPERATION NAME</th><th>PERFORMANCE / USAGE</th></tr>
<tr><td>sendInsuranceData()</td><td>Required response time: N/A
Maximal size of message (request/response): N/A
Average size of message (request/response): N/A
Maximal count of message for peak: 5/s
Maximal count of message for day: N/A</td></tr>
</table>
</td></tr>
</table>

4 Covered scenarios

4.1 Common information

The SOAP service is described using WSDL file, which has documentation for each field. So duplicating that documentation to this file is not necessary.

4.1.1 Request header

Request header contains state information of the message transmitted. Each field is described in SeSp documentation. The values do not have any business meaning but are present to be able to track each message.

4.1.1.1 Example request header

```
<urn1:header>
    <urn1:timestamp>2023-03-01T12:25:41</urn1:timestamp>
    <urn1:messageId>09665ac3-c4da-4d56-8e10-fa6414bdb8de</urn1:messageId>
    <urn1:correlationId>d8309ffb-55a3-4f3e-b3d0-2e8d729b7db8</urn1:correlationId>
    <urn1:sender>INSURANCE_COMPANY_NAME</urn1:sender>
    <urn1:receiver>PGRLF</urn1:receiver>
</urn1:header>
```

4.1.2 Response header

Response header is subclass of request header and contains state information of the message transmitted. The header has two more fields to specify message processing status (OK/ERROR/…). Each field is described in SeSp documentation. Most of the values do not have any business meaning (status, message), but are present to be able to track each message.

4.1.2.1 Example response header

```
<urn1:header>
   <urn1:timestamp>2023-03-01T12:25:41</urn1:timestamp>
   <urn1:messageId>6089d99e-4b3c-4f6b-9ec3-6eeb9ff99f35</urn1:messageId>
   <urn1:correlationId>d8309ffb-55a3-4f3e-b3d0-2e8d729b7db8</urn1:correlationId>
   <urn1:sender>PGRLF</urn1:sender>
   <urn1:receiver>INSURANCE_COMPANY_NAME</urn1:receiver>
   <urn1:status>OK</urn1:status>
</urn1:header>
```

4.1.3 Error handling

Error states will be transmitted in response header in fields status and message. Status will contain ERROR and the message will contain detailed information of that error.

4.2 Insurance fully paid – finished

This request should be sent only if:

1. All planned payments are paid,
2. The date of the last planned payment is due,
3. Each planned payment has mapped payment details and the planned amount is sum of payment details on that planned payment.

4.2.1 Request restrictions

1. Do not send partial information using this interface, e.g. client paid the first planned payment, but the rest is still not paid.
2. Do not send the same request repeatedly – same message id.

3. One request can contain multiple insurance information, not only one. But it is preferred to send one SOAP message per client (policyholder) for security concerns.

4.2.2 Example request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:urn="urn:cz:pgrlf:subsidy:insurance:insurance-subsidy:v1.0"
                  xmlns:urn1="urn:cz:pgrlf:common:v1.0">
    <soapenv:Header/>
    <soapenv:Body>
        <urn:InsuranceDataRequest>
            <urn1:header>
                <urn1:timestamp>2023-03-01T12:25:41</urn1:timestamp>
                <urn1:messageId>09665ac3-c4da-4d56-8e10-fa6414bdb8de</urn1:messageId>
                <urn1:correlationId>d8309ffb-55a3-4f3e-b3d0-2e8d729b7db8</urn1:correlationId>
                <urn1:sender>INSURANCE_COMPANY_NAME</urn1:sender>
                <urn1:receiver>PGRLF</urn1:receiver>
            </urn1:header>
            <urn:body>
                <urn:insurance>
                    <urn:number>123456789</urn:number>
                    <urn:subprogram>Z</urn:subprogram>
                    <urn:contractDate>2023-03-01</urn:contractDate>
                    <urn:startDate>2023-03-01</urn:startDate>
                    <urn:policyholder>Some policyholder name, a.s.</urn:policyholder>
                    <urn:ic>123456789</urn:ic>
                    <urn:updatedDate>2023-03-01</urn:updatedDate>
                    <urn:amount>2000</urn:amount>
                    <urn:riskAmount>2000</urn:riskAmount>
                    <urn:currency>CZK</urn:currency>
                    <urn:paymentInterval>ANNUALLY</urn:paymentInterval>
                    <urn:amountPaid>2000</urn:amountPaid>
                    <urn:riskAmountPaid>2000</urn:riskAmountPaid>
                    <urn:payment>
                        <urn:index>1</urn:index>
                        <urn:amount>2000</urn:amount>
                        <urn:currency>CZK</urn:currency>
                        <urn:plannedDate>2023-03-01</urn:plannedDate>
                        <urn:paymentDetail>
                            <urn:amount>1000</urn:amount>
                            <urn:currency>CZK</urn:currency>
                            <urn:date>2023-03-01</urn:date>
                            <urn:source>PROPER_PAYMENT</urn:source>
                        </urn:paymentDetail>
                        <urn:paymentDetail>
                            <urn:amount>1000</urn:amount>
                            <urn:currency>CZK</urn:currency>
                            <urn:date>2023-03-01</urn:date>
                            <urn:source>OVERPAYMENT_DEINSURANCE</urn:source>
                        </urn:paymentDetail>
                    </urn:payment>
                    <urn:paymentSource>CLIENT</urn:paymentSource>
                    <urn:state>FINISHED</urn:state>
                    <urn:message></urn:message>
                </urn:insurance>
            </urn:body>
        </urn:InsuranceDataRequest>
    </soapenv:Body>
</soapenv:Envelope>
```

4.2.3 Example response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:cz:pgrlf:subsidy:insurance:insurance-subsidy:v1.0"
xmlns:urn1="urn:cz:pgrlf:common:v1.0">
   <soapenv:Header/>
   <soapenv:Body>
      <urn:InsuranceDataResponse>
         <urn1:header>
            <urn1:timestamp>2023-03-01T12:25:41</urn1:timestamp>
            <urn1:messageId>6089d99e-4b3c-4f6b-9ec3-6eeb9ff99f35</urn1:messageId>
            <urn1:correlationId>d8309ffb-55a3-4f3e-b3d0-2e8d729b7db8</urn1:correlationId>
            <urn1:sender>PGRLF</urn1:sender>
            <urn1:receiver>INSURANCE_COMPANY_NAME</urn1:receiver>
            <urn1:status>OK</urn1:status>
```

```
        </urn1:header>
        <urn:body/>
      </urn:InsuranceDataResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

4.3 Insurance cancelled

This request should be sent only if:

1. Client cancelled the insurance.

4.3.1 Request restrictions

1. All the planned payments elements should be present. If the amount paid subtracted by the amount returned to the client does not cover all the planned payments, then the planned payments payment details don't have to be present, or don't have to sum up to the planned payment amount.
2. State field must be set to CANCELLED.
3. Message field should contain more detailed information of what happened (Client cancelled because …)
4. All other fields shall be filled as for scenario 4.2.

4.3.2 Example request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:urn="urn:cz:pgrlf:subsidy:insurance:insurance-subsidy:v1.0"
                  xmlns:urn1="urn:cz:pgrlf:common:v1.0">
    <soapenv:Header/>
    <soapenv:Body>
        <urn:InsuranceDataRequest>
            <urn1:header>
                <urn1:timestamp>2023-03-01T12:33:14</urn1:timestamp>
                <urn1:messageId>34075c69-dfec-450c-808f-d277283693ff</urn1:messageId>
                <urn1:correlationId>d8309ffb-55a3-4f3e-b3d0-2e8d729b7db8</urn1:correlationId>
                <urn1:sender>INSURANCE_COMPANY_NAME</urn1:sender>
                <urn1:receiver>PGRLF</urn1:receiver>
            </urn1:header>
            <urn:body>
                <urn:insurance>
                    <urn:number>123456789</urn:number>
                    <urn:subprogram>Z</urn:subprogram>
                    <urn:contractDate>2023-03-01</urn:contractDate>
                    <urn:startDate>2023-03-01</urn:startDate>
                    <urn:policyholder>Some policyholder name, a.s.</urn:policyholder>
                    <urn:ic>123456789</urn:ic>
                    <urn:updatedDate>2023-03-01</urn:updatedDate>
                    <urn:amount>2000</urn:amount>
                    <urn:riskAmount>2000</urn:riskAmount>
                    <urn:currency>CZK</urn:currency>
                    <urn:paymentInterval>ANNUALLY</urn:paymentInterval>
                    <urn:amountPaid>1000</urn:amountPaid>
                    <urn:riskAmountPaid>2000</urn:riskAmountPaid>
                    <urn:payment>
                        <urn:index>1</urn:index>
                        <urn:amount>2000</urn:amount>
                        <urn:currency>CZK</urn:currency>
                        <urn:plannedDate>2023-03-01</urn:plannedDate>
                        <urn:paymentDetail>
                            <urn:amount>1000</urn:amount>
                            <urn:currency>CZK</urn:currency>
                            <urn:date>2023-03-01</urn:date>
                            <urn:source>OVERPAYMENT_DEINSURANCE</urn:source>
                        </urn:paymentDetail>
                    </urn:payment>
                    <urn:paymentSource>CLIENT</urn:paymentSource>
                    <urn:state>CANCELLED</urn:state>
                    <urn:message>Customer cancelled the insurance</urn:message>
                </urn:insurance>
            </urn:body>
```

```
        </urn:InsuranceDataRequest>
    </soapenv:Body>
</soapenv:Envelope>
```

4.3.3 Example response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:cz:pgrlf:subsidy:insurance:insurance-subsidy:v1.0"
xmlns:urn1="urn:cz:pgrlf:common:v1.0">
   <soapenv:Header/>
   <soapenv:Body>
      <urn:InsuranceDataResponse>
         <urn1:header>
            <urn1:timestamp>2023-03-01T12:25:41</urn1:timestamp>
            <urn1:messageId>6089d99e-4b3c-4f6b-9ec3-6eeb9ff99f35</urn1:messageId>
            <urn1:correlationId>d8309ffb-55a3-4f3e-b3d0-2e8d729b7db8</urn1:correlationId>
            <urn1:sender>PGRLF</urn1:sender>
            <urn1:receiver>INSURANCE_COMPANY_NAME</urn1:receiver>
            <urn1:status>OK</urn1:status>
         </urn1:header>
         <urn:body/>
      </urn:InsuranceDataResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

5 Connection

To connect to a SOAP service hosted on Azure API Management (APIM) you will need few information:

1. API endpoint URL: **Will be sent to consumer when available.**
2. WSDL file
    - Can be downloaded from the API endpoint URL,
    - Will be distributed with this documentation.
3. API endpoint authentication method: **Client certificate authentication**
    - It is possible to enable one of the less secure authentication methods for development phase.
4. API credentials: **Will be sent according to chapter 6.2**.
5. Communication protocol: **Strictly https**
6. Allowed IP addresses: **IP address of connected insurance company**

6 Security

6.1 Authentication

Service will support multiple authentication methods:

1. OAuth2 with client certificate authentication - Preferred form of authentication
2. Client Certificate authentication
3. API Key authentication - fallback method, if client is not able to authenticate using above methods
4. Basic Authentication - fallback method, if client is not able to authenticate using above methods

6.1.1 OAuth2 with client certificate

To authenticate to a SOAP service on Azure using OAuth2 and client certificates, you can follow these steps:

1. Register your application in Azure Active Directory (AD) and obtain a client ID and client secret.
2. Configure the SOAP service to require client certificate authentication.
3. *Create a client certificate and upload it to Azure AD.*
    - *Consumer must provide certificate.*
4. Configure the Azure AD application to use the client certificate for authentication.
5. Obtain an access token from Azure AD using OAuth2 authentication and the client ID and client secret.
6. Add the access token to the SOAP request headers.

6.1.1.1 Detail description

1. Register your application in Azure AD:
    - Go to the Azure portal and select "Azure Active Directory" from the left-hand menu.
    - Select "App registrations" and click "New registration".
    - Enter a name for your application and select the appropriate account types.
    - Under "Redirect URI", select "Web" and enter a URL where the authorization code will be sent after the user authenticates.
    - Click "Register" to create the application and obtain the client ID and client secret.
2. Configure the SOAP service to require client certificate authentication:
    - Depending on the SOAP service, this step may vary. Consult the documentation for your specific service for instructions.
3. Create a client certificate and upload it to Azure AD
    - Create a self-signed certificate or obtain one from a trusted certificate authority.
    - In Azure AD, select "Certificates & secrets" and click "New client secret".
    - Enter a description for the certificate and upload the .cer file.
4. Create a self-signed certificate or obtain one from a trusted certificate authority.
    - In Azure AD, select "Certificates & secrets" and click "New client secret".
    - Enter a description for the certificate and upload the .cer file.
    - Select "Manifest" and add the following JSON to the "appRoles" property:

```
{
    "allowedMemberTypes": [
      "Application"
    ],
    "displayName": "Access SOAP service with client certificate",
    "id": "<GUID>",
    "isEnabled": true,
    "description": "Allows the application to access the SOAP service with a client
certificate.",
```

```
    "value": "access_soap_service_with_client_certificate"
}
```

- Replace <GUID> with a unique GUID.
- Save the manifest changes.

5. Obtain an access token from Azure AD using OAuth2 authentication and the client ID and client secret:
   - Send a POST request to the Azure AD token endpoint (https://login.microsoftonline.com/{tenant}/oauth2/token) with the following parameters:
     - **grant_type**: client_credentials
     - **client_id**: {your_client_id}
     - **client_secret**: {your_client_secret}
     - **resource**: {the_resource_URI_of_the_SOAP_service}
     - **client_assertion_type**: urn:ietf:params:oauth:client-assertion-type:jwt-bearer
     - **client_assertion**: {a_JWT_Bearer_token_signed_with_the_client_certificate}
   - The response will contain an access token.
6. Add the access token to the SOAP request headers:
   - Include the access token in the SOAP request headers with the "Authorization" header, like this:

     Authorization: Bearer {access_token}

   - Send the SOAP request with the headers included.

6.1.2 Client certificate

To authenticate to a SOAP service on Azure using client certificates, you can follow these steps:

1. *Upload the client certificate to APIM.*
   - *Consumer must provide certificate.*
2. Create an inbound policy in APIM to authenticate requests using the client certificate.
3. Add the policy to the SOAP API in APIM.

6.1.2.1 Detail description

1. Upload the client certificate to APIM:
   - Go to the APIM portal and select your API.
   - Select "Security" from the left-hand menu.
   - Under "Certificates", click "Upload".
   - Upload the client certificate in .cer
2. Create an inbound policy in APIM to authenticate requests using the client certificate:
   - In the APIM portal, select "APIs" from the left-hand menu and select your API.
   - Select "Policies" from the left-hand menu.
   - Add the following policy to the inbound section

     ```
     <inbound>
       <authentication-certificate thumbprint="{certificate_thumbprint}" />
     </inbound>
     ```

   - Replace {certificate_thumbprint} with the thumbprint of the uploaded certificate.
3. Add the policy to the SOAP API in APIM:
   - In the APIM portal, select your API.
   - Select "Operations" from the left-hand menu.
   - Select the SOAP operation to which you want to add the policy.
   - Select "Add policy" and select the "Inbound" option.
   - Add the policy you created in step 2.
4. Test the policy by sending a SOAP request with the client certificate:
   - Send a SOAP request to the API endpoint using a tool such as SOAP UI.
   - Make sure to include the client certificate in the request.

- If the policy is working correctly, the request should be authenticated and the SOAP response should be returned.

6.1.3 API Key

To authenticate to a SOAP service on Azure using API key, you can follow these steps:

1. Create an API key in APIM.
2. Add the API key to the SOAP API in APIM.
3. *Test the API key by sending a SOAP request with the API key.*

6.1.3.1 Detail description

1. Create an API key in APIM:
   - In the APIM portal, select your API.
   - Select "Security" from the left-hand menu.
   - Under "Client certificates and API keys", select "Add".
   - Enter a name for the API key and select the appropriate options for the key.
   - Click "Create" to generate the API key.
2. Add the API key to the SOAP API in APIM:
   - In the APIM portal, select your API.
   - Select "Operations" from the left-hand menu.
   - Select the SOAP operation to which you want to add the API key.
   - Select "Add policy" and select the "Inbound" option.
   - Add the following policy to the inbound section:

```
<inbound>
    <base/>
    <set-header name="Ocp-Apim-Subscription-Key" exists-action="override">
        <value>{API_key_value}</value>
    </set-header>
</inbound>
```

   - Replace {API_key_value} with the value of the API key you created in step 1.
3. Test the API key by sending a SOAP request with the API key:
   - Send a SOAP request to the API endpoint using a tool such as SOAP UI.
   - Make sure to include the API key in the request headers with the key name "Ocp-Apim-Subscription-Key".
   - If the API key is working correctly, the request should be authenticated and the SOAP response should be returned.

6.1.4 Basic authentication

To authenticate to a SOAP service on Azure using basic authentication, you can follow these steps:

1. Create a user in APIM.
2. Add the user to the SOAP API in APIM.
3. *Test the basic authentication by sending a SOAP request with the user credentials.*

6.1.4.1 Detail description

1. Create a user in APIM:
   - In the APIM portal, select "Users" from the left-hand menu.
   - Select "Add" to create a new user.
   - Enter the user's details and select the appropriate options for the user.
   - Click "Create" to generate the user.
2. Add the user to the SOAP API in APIM:
   - In the APIM portal, select your API.
   - Select "Security" from the left-hand menu.
   - Under "Basic authentication", select "Add user".

- Enter the username and password for the user you created in step 1.

3. Test the basic authentication by sending a SOAP request with the user credentials:
   - Send a SOAP request to the API endpoint using a tool such as SOAP UI.
   - Make sure to include the user credentials in the request headers using the "Authorization" header with the value "Basic {base64-encoded-credentials}".
   - Replace "{base64-encoded-credentials}" with the base64-encoded string of the format "username:password".
   - If the basic authentication is working correctly, the request should be authenticated, and the SOAP response should be returned.

6.2 Credentials handover

It is important to ensure that the credentials are transmitted securely to prevent it from being intercepted or tampered with. The credentials are one of or multiple of:

1. Client id
2. Client secret
3. Public part of client certificate
4. API key
5. Username
6. Password

Following steps will be taken to handover the credentials:

1. The credentials will be crypted using password, 16 chars long, including lowercase and uppercase characters, numbers and special symbols,
2. Send the encrypted credential to agreed email address,
   a. It is almost impossible to send unencrypted **.cer** files over email service.
3. Send the encryption key (password) to agreed phone number.